
OTUS
ОНЛАЙН-ОБРАЗОВАНИЕ
Онлайн-образование

Меня хорошо видно && слышно?

Ставьте  , если все хорошо

Напишите в чат, если есть проблемы

Проверить, идет ли запись!

Nomad

Кузин Евгений

Преподаватель

Кузин Евгений

15 лет опыта работы с linux, сетями и безопасностью.
Ведущий инженер в компании Revolut

Правила вебинара

Активно участвуем

Задаем вопросы в чат или голосом

Off-topic обсуждаем в Slack #канал группы или #general

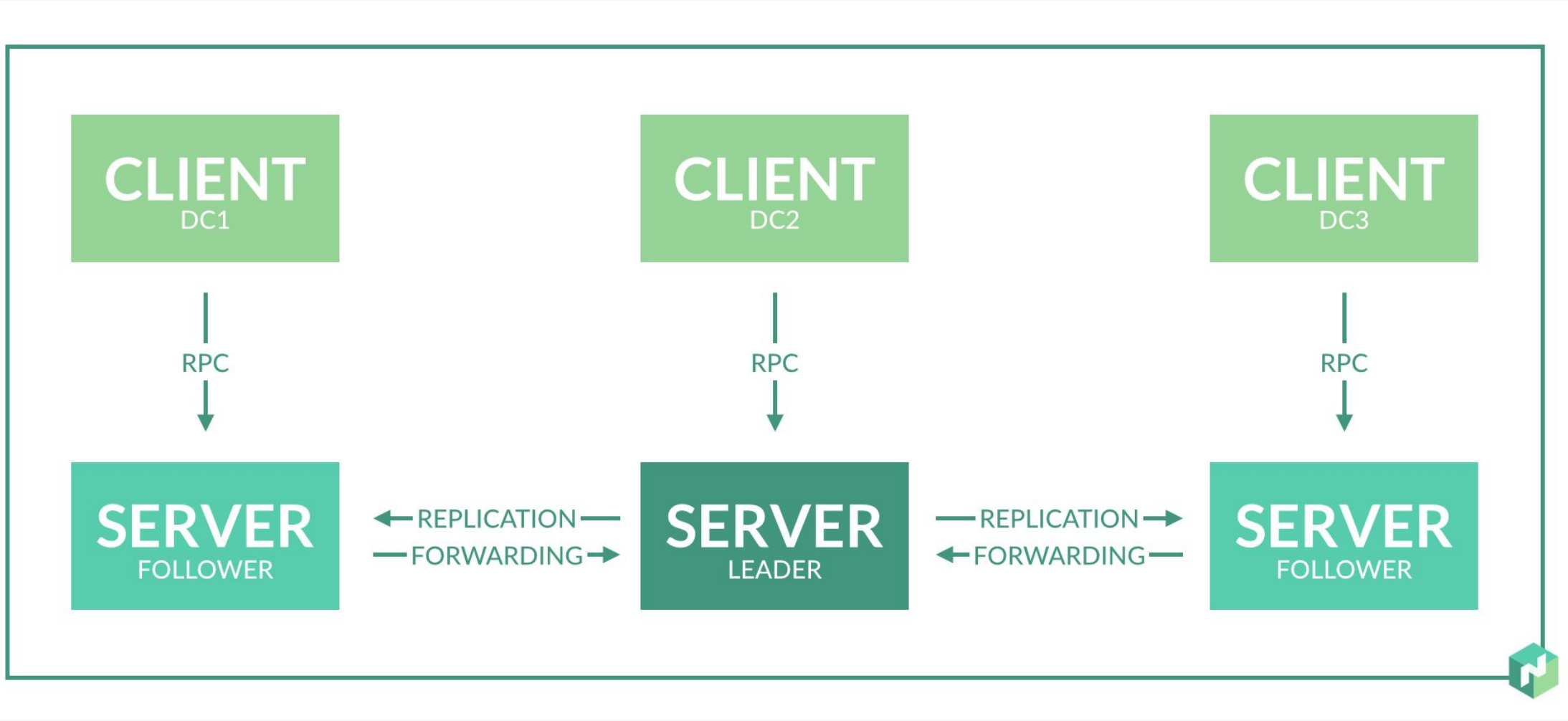
Вопросы вижу в чате, могу ответить не сразу

Маршрут вебинара

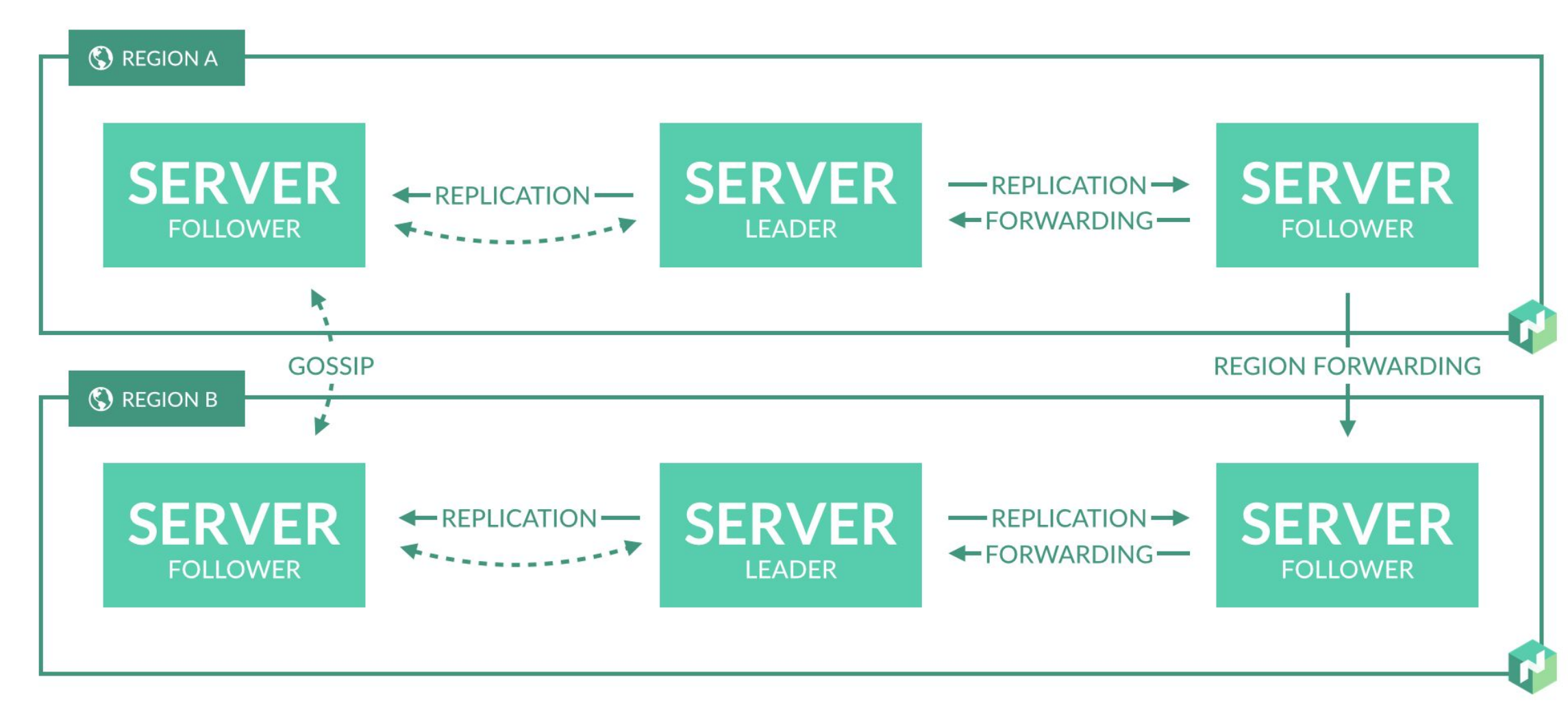
Glossary

- **Job** - A Job is a form of *desired state*; the user is expressing that the job should be running, but not where it should be run.
- **Task Group** - A Task Group is a set of tasks that must be run together.
- **Driver** – A Driver represents the basic means of executing your **Tasks**. Example Drivers include Docker, QEMU, Java, and static binaries.
- **Task** - A Task is the smallest unit of work in Nomad. Tasks are executed by drivers,
- **Client** - A Client of Nomad is a machine that tasks can be run on. All clients run the Nomad agent.
- **Allocation** - An Allocation is a mapping between a task group in a job and a client node.
- **Evaluation** - Evaluations are the mechanism by which Nomad makes scheduling decisions.
- **Server** - Nomad servers are the brains of the cluster. Can do HA and Federate.
- **Regions and Datacenters** - Nomad models infrastructure as regions and datacenters. Regions may contain multiple datacenters.
- **Bin Packing** - Bin Packing is the process of filling bins with items in a way that maximizes the utilization of bins. This extends to Nomad, where the clients are "bins" and the items are task groups. Nomad optimizes resources by efficiently bin packing tasks onto client machines.

Overview


CLIENT
DC1
CLIENT
DC2
CLIENT
DC3
RPC
RPC
RPC
SERVER
FOLLOWER
REPLICATION
FORWARDING
SERVER
LEADER
REPLICATION
FORWARDING
SERVER
FOLLOWER

Overview


REGION A
SERVER
FOLLOWER
REPLICATION
SERVER
LEADER
REPLICATION
FORWARDING
SERVER
FOLLOWER
GOSSIP
REGION FORWARDING
REGION B
SERVER
FOLLOWER
REPLICATION
SERVER
LEADER
REPLICATION
FORWARDING
SERVER
FOLLOWER

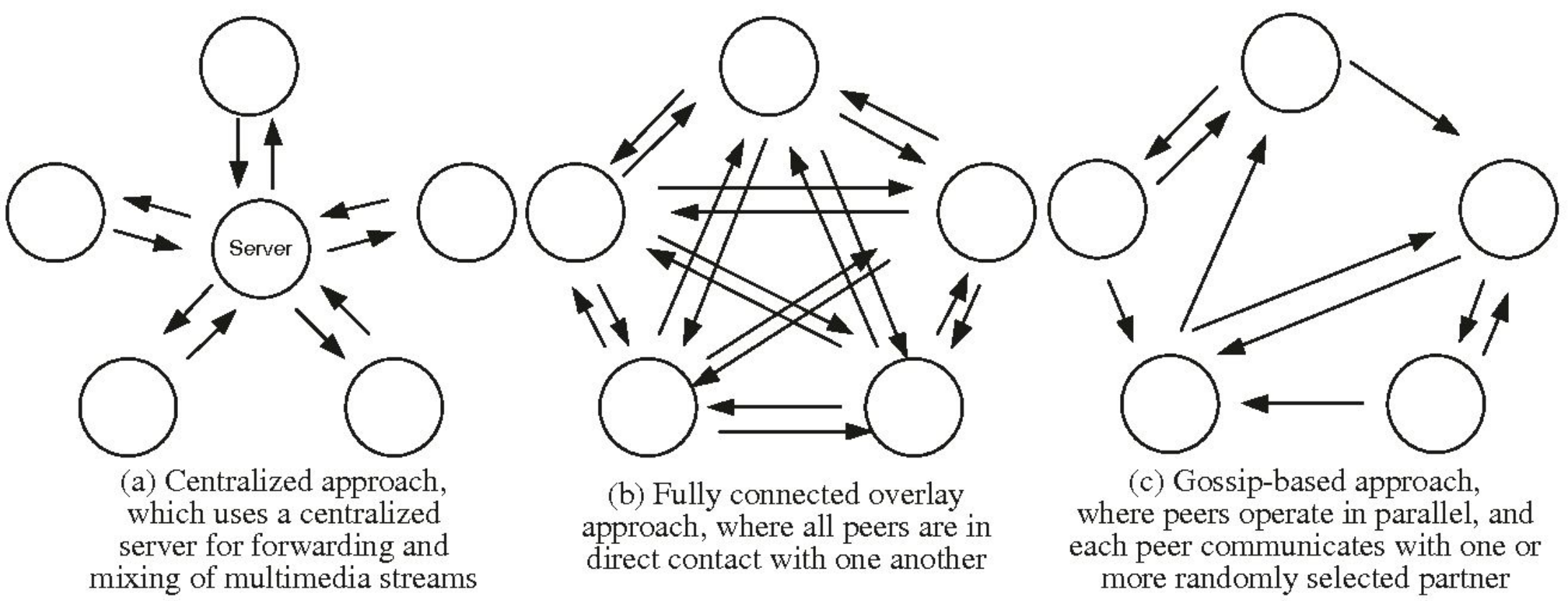
Consensus protocol

Raft

- **Log** - The primary unit of work in a Raft system is a log entry. The problem of consistency can be decomposed into a *replicated log*.
- **FSM** - Finite State Machine. Application of the same sequence of logs must result in the same state, meaning behavior must be deterministic.
- **Peer set** - The peer set is the set of all members participating in log replication.
- **Quorum** - A quorum is a majority of members from a peer set: for a set of size n, quorum requires at least $\lfloor(n/2)+1\rfloor$ members.
- **Committed Entry** - An entry is considered *committed* when it is durably stored on a quorum of nodes. Once an entry is committed it can be applied.
- **Leader** - At any given time, the peer set elects a single node to be the leader. The leader is responsible for ingesting new log entries, replicating to followers, and managing when an entry is considered committed.

Gossip protocol

Gossip protocol

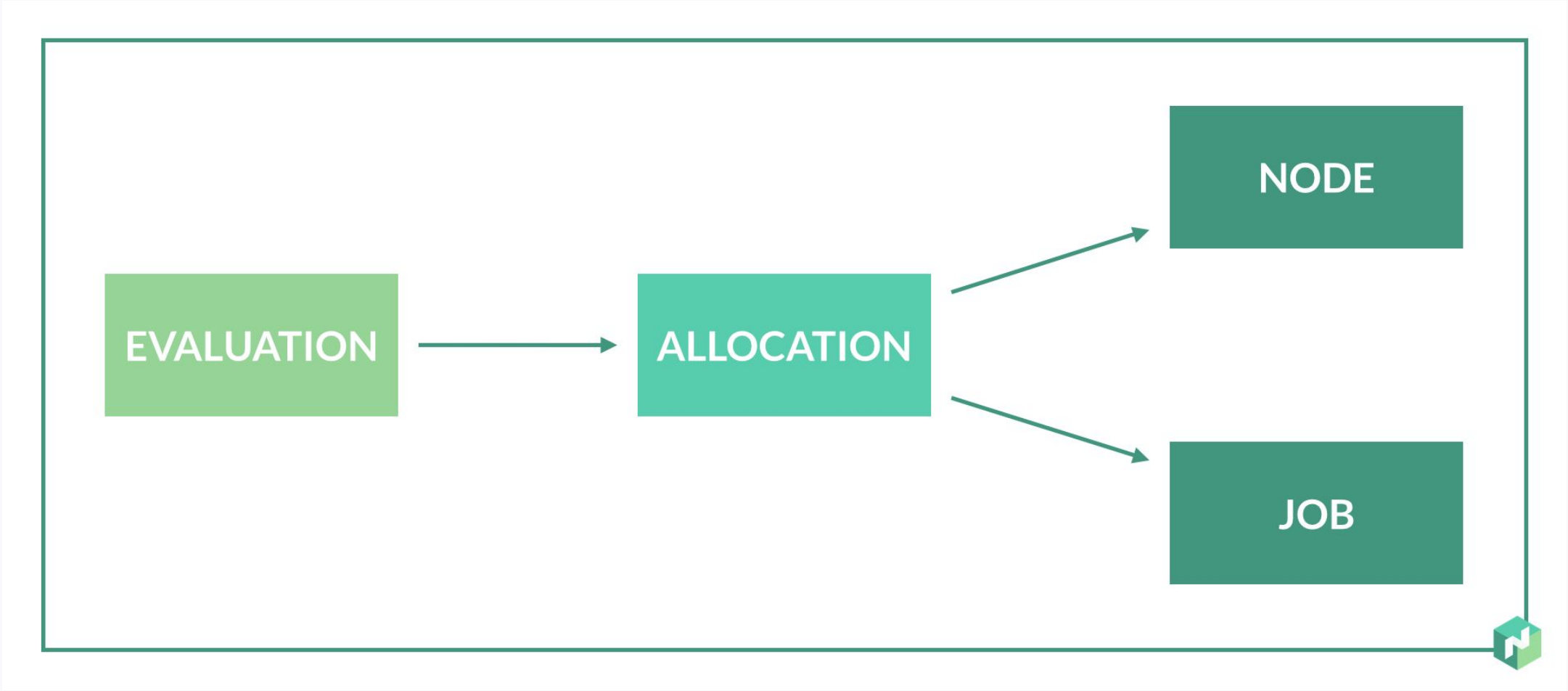


(a) Centralized approach, which uses a centralized server for forwarding and mixing of multimedia streams

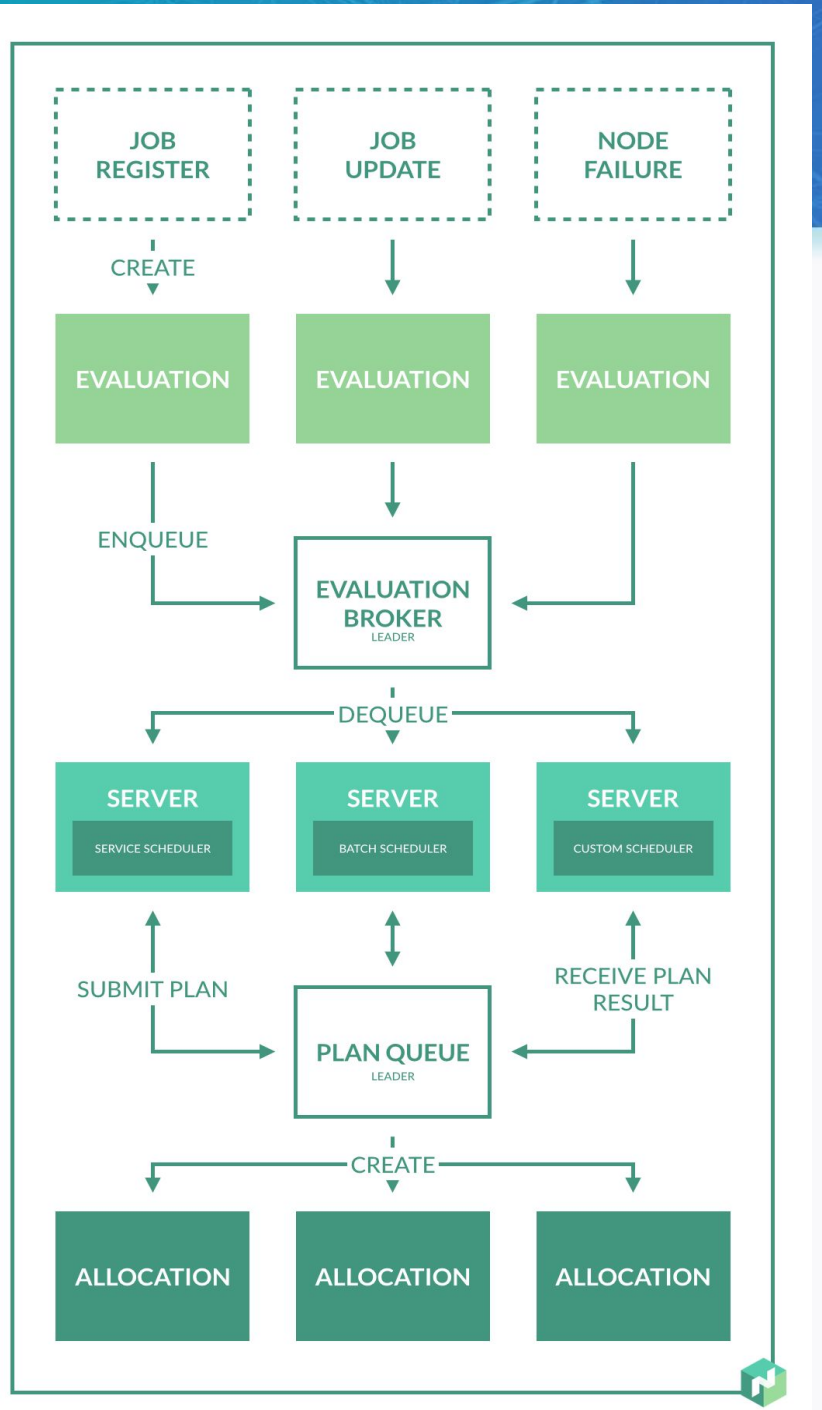
(b) Fully connected overlay approach, where all peers are in direct contact with one another

(c) Gossip-based approach, where peers operate in parallel, and each peer communicates with one or more randomly selected partner

Scheduling



Scheduling


JOB REGISTER
JOB UPDATE
NODE FAILURE
CREATE
EVALUATION
EVALUATION
EVALUATION
ENQUEUE
EVALUATION BROKER
LEADER
DEQUEUE
SERVER
SERVICE SCHEDULER
SERVER
BATCH SCHEDULER
SERVER
CUSTOM SCHEDULER
SUBMIT PLAN
RECEIVE PLAN RESULT
PLAN QUEUE
LEADER
CREATE
ALLOCATION
ALLOCATION
ALLOCATION

Preemption

Что если все ресурсы закончились, а нам ну очень надо задеплоить еще?

Job Priority!

Job	Priority	Allocations	Total Used capacity
cache	70	a6	2 GB Memory, 0.5 GB Disk, 1 CPU
batch-analytics	50	a4, a5	<1 GB Memory, 0.5 GB Disk, 0.5 CPU>, <1 GB Memory, 0.5 GB Disk, 0.5 CPU>
email-marketing	20	a1, a2	<0.5 GB Memory, 0.8 GB Disk>, <0.5 GB Memory, 0.2 GB Disk>

Task Drivers

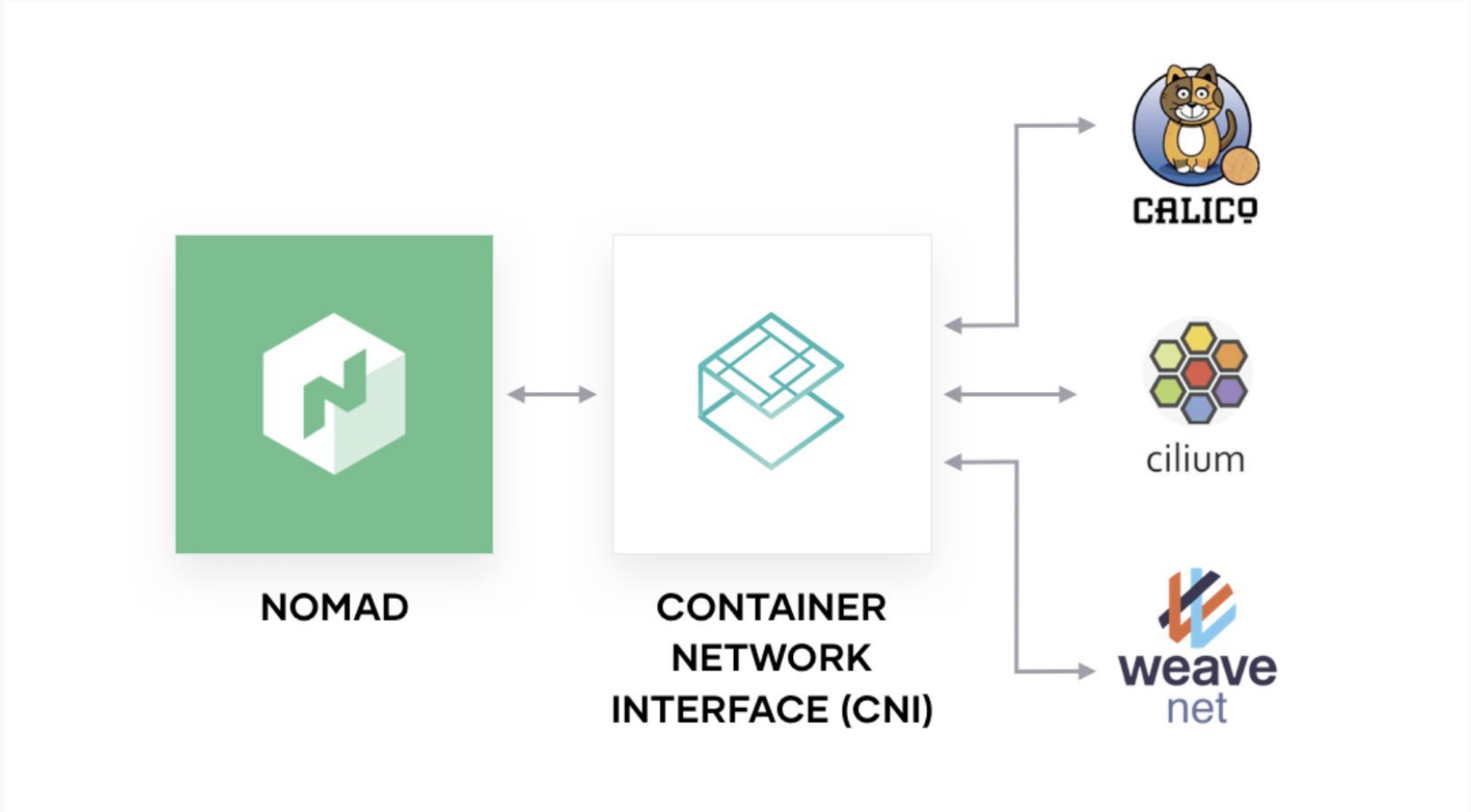
- Docker
- Isolated Fork/Exec
- Java
- Podman
- QEMU
- Raw Fork/Exec

И еще много других поддерживаемых community:
[Task Driver Plugins: Community Supported]

Device Drivers

Nvidia

CNI



[Multi-Interface Networking and CNI Plugins in Nomad]

Filesystem

```
.
├── alloc  Общая директория для всех заданий
│   ├── data  ephemeral_disk будут тут
│   ├── logs     тут лежат логи, которые можно посмотреть через nomad alloc logs
│   │   ├── task1.stderr.0
│   │   ├── task1.stdout.0
│   │   ├── task2.stderr.0
│   │   └── task2.stdout.0
│   └── tmp
├── task1
│   ├── local     NOMAD_TASK_DIR
│   ├── secrets   NOMAD_SECRETS_DIR
│   └── tmp
└── task2
    ├── local
    ├── secrets
    └── tmp
```

Filesystem isolation (docker & qemu)

```
$ docker inspect 32e | jq '.[0].HostConfig.Binds'
[
  "/var/nomad/alloc/b0686b27-8af3-8252-028f-af485c81a8b3/alloc:/alloc",
  "/var/nomad/alloc/b0686b27-8af3-8252-028f-af485c81a8b3/task1/local:/local",
  "/var/nomad/alloc/b0686b27-8af3-8252-028f-af485c81a8b3/task1/secrets:/secrets"
]
$ docker exec -it 32e /bin/sh
# ls /
alloc  boot  dev  home  lib64  media  opt   root  sbin     srv  tmp  var
bin    data  etc  lib   local  mnt    proc  run   secrets  sys  usr
```

Filesystem isolation (exec & java)

используется chmod & pivot_root для изоляции процесса.
Следующие элементы будут скопированы в новое пространство (список можно поменять с помощью опции chroot_env)

```
[
  "/bin",
  "/etc",
  "/lib",
  "/lib32",
  "/lib64",
  "/run/resolvconf",
  "/sbin",
  "/usr",
    ]
```

```
.
├── alloc
│   ├── container
│   ├── data
│   ├── logs
│   └── tmp
└── task2
    ├── alloc
    ├── bin
    ├── dev
    ├── etc
    ├── executor.out
    ├── lib
    ├── lib32
    ├── lib64
    ├── local
    ├── proc
    ├── run
    ├── sbin
    ├── secrets
    ├── sys
    ├── tmp
    └── usr
```

Filesystem isolation (none)

Никакой изоляции вообще.
Полная свобода и права на машине как у пользователя под которым запущен nomad

```
$ nomad alloc exec 87ec7d12 /bin/sh
# ls /
bin   dev  home        lib    lib64   lost+found  mnt  proc  run   snap  sys  usr
vmlinuz
boot  etc  initrd.img  lib32  libx32  media       opt  root  sbin  srv   tmp  var

# echo $NOMAD_SECRETS_DIR
/var/nomad/alloc/87ec7d12-5e35-8fba-96cc-09e5376be15a/task3/secrets

# whoami
root
```

Filesystem isolation (none)

Как происходит аллокация файловой системы для задачи:

- Новая директория для аллокации
- ephemeral_disk копируется с предыдущей ноды если возможно
- Выполняются CSI разделы
- Для каждой **Task**:
  - Новая директория для **Task**
  - копируются Dispatch payloads
  - скачиваются Artifacts
  - генерируются Templates
  - **Task** запускается с выбранным драйвером и всеми volume mounts

Security

- **`mTLS`**
- **`ACL`**

Port / Protocol	Agents	Description
4646 / TCP	All	HTTP для UI и API
4647 / TCP	Servers	RPC
4648 / TCP + UDP	Servers	gossip

Requirements

- 10ms внутри дата центра
- 3-5 серверов на регион
- 100+ ms от клиента к серверу
- для сервера нужно только папка на диске и возможность открывать сетевые соединения
- для агента нужен root

Когда задачи запрашивают динамические порты, они выделяются из диапазона портов от 20 000 до 32 000.
проверить и при необходимости выставить новые значения для Linux можно так:

```
$ cat /proc/sys/net/ipv4/ip_local_port_range
32768   60999
$ echo "49152 65535" > /proc/sys/net/ipv4/ip_local_port_range
```

LAB

```
nomad job init
```

HowTo

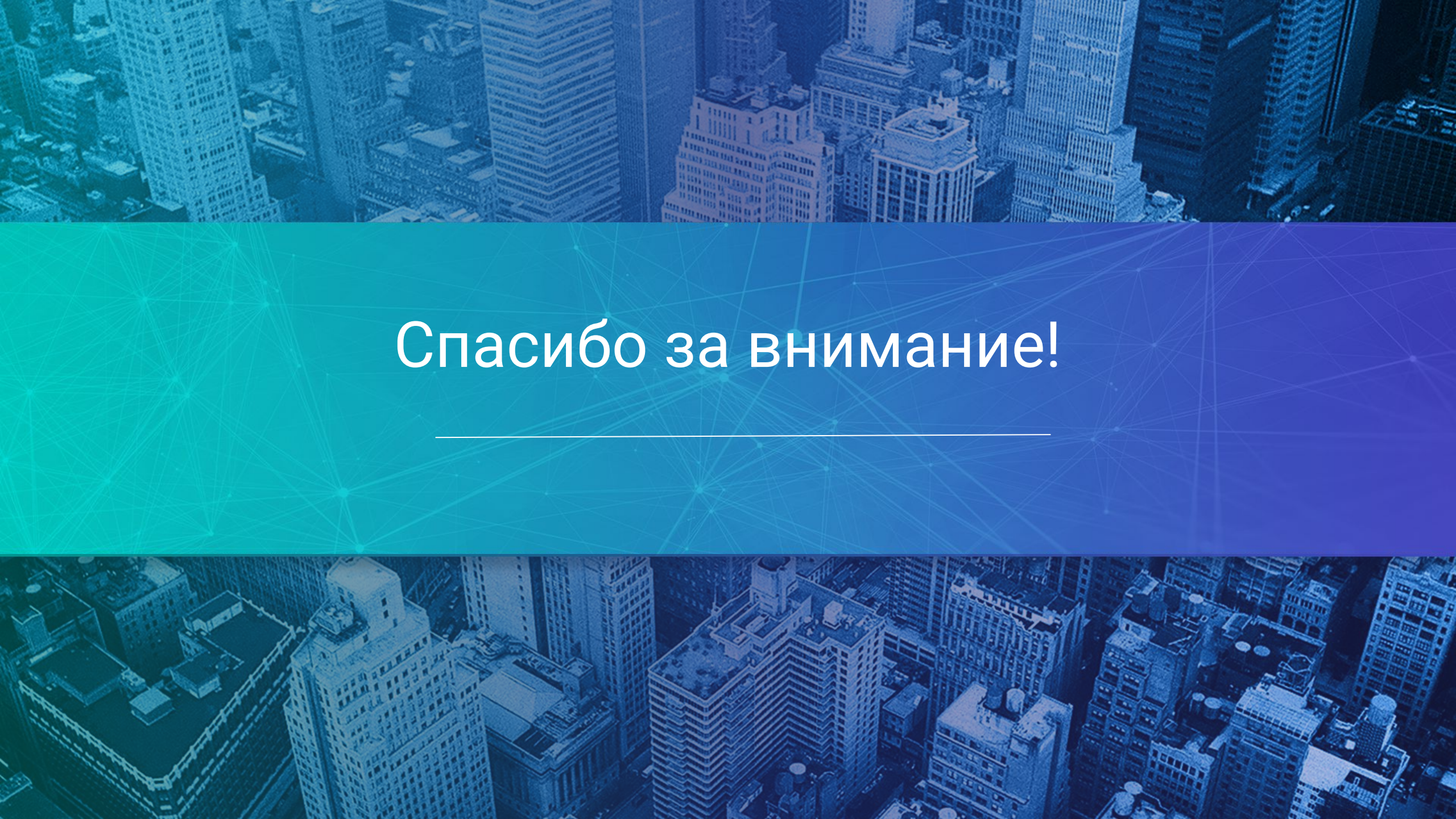
Autopilot | Nomad

Configure NGINX Reverse Proxy for Nomad's Web UI | Nomad

Test Drive the Nomad Autoscaler with Vagrant | Nomad

Load Balancing with NGINX | Nomad

Outage Recovery | Nomad


Спасибо за внимание!